# Exploiting effective negative curvature directions via SYMMBK algorithm, in Newton–Krylov methods

Giovanni Fasano[1], Christian Piermarini[2], Massimo Roma[2*]

[1]Dep. of Management, University Ca' Foscari,
San Giobbe Cannaregio 873, Venezia, 30121, Italy.
[2]Dep. of Computer, Control and Management Engineering "A. Ruberti",
SAPIENZA University of Rome, via Ariosto 25, Roma, 00185, Italy.

*Corresponding author(s). E-mail(s): roma@diag.uniroma1.it;
Contributing authors: fasano@unive.it; piermarini@diag.uniroma1.it;

**Abstract**

In this paper we consider the issue of computing negative curvature directions, for nonconvex functions, within Newton–Krylov methods for large scale unconstrained optimization. In the last decades this issue has been widely investigated in the literature, and different approaches have been proposed. We focus on the well known SYMMBK method introduced for solving large scale symmetric possibly indefinite linear systems [5, 9, 11, 28], and show how to exploit it to yield an effective negative curvature direction in optimization frameworks. The distinguishing feature of our proposal is that the computation of negative curvatures is basically carried out as by–product of SYMMBK procedure, without storing no more than one additional vector. Hence, no explicit matrix factorization or matrix storage is required. An extensive numerical experimentation has been performed on CUTEst problems; the obtained results have been analyzed also through novel profiles (*Quality Profiles*) which highlighted the good capability of the algorithms which use negative curvature directions to determine better local minimizers.

**Keywords:** Large scale unconstrained optimization, Newton–Krylov methods, Negative curvature directions, Second order critical points, Quality profiles

# 1 Introduction

We focus on linesearch–based Newton–Krylov methods that are widely used for solving large scale unconstrained optimization problems, namely to determine a local minimizer of the problem:

$$\min_{x\in\mathbb{R}^n} f(x), \tag{1.1}$$

being $f : \mathbb{R}^n \to \mathbb{R}$ a twice continuously differentiable function. Given an initial guess $x_0 \in \mathbb{R}^n$, at each iteration of these methods, a new iterate is generated according to the iterative scheme

$$x_{k+1} = x_k + \alpha_k d_k, \tag{1.2}$$

where $d_k$ is a search direction and $\alpha_k > 0$ is a suited steplength. Since the search direction is determined by means of an iterative Krylov–subspace method, a linesearch–based scheme for a Newton–Krylov method encompasses two nested loops:

- the *outer iterations*, where starting from the current iterate $x_k$, a new iterate is generated according to the scheme (1.2), and where $\alpha_k$ is computed by a linesearch procedure;
- the *inner iterations*, namely the iterations of the Krylov–subspace method used for approximately solving the Newton equation

$$\nabla^2 f(x_k)d = -\nabla f(x_k), \tag{1.3}$$

at each outer iteration $k$. Newton–Krylov methods are also called Truncated Newton methods (or inexact Newton methods) since the inner iterations are usually "truncated", i.e. terminated according to a suited stopping criterion, still ensuring superlinear converge rate [16, 17]. As concerns global convergence properties, convergence to first order critical points is guaranteed, i.e. points which satisfies first order necessary optimality conditions, namely towards stationary points. For a complete overview of these methods we refer the reader to the survey paper by Nash [33].

Among the most commonly used iterative methods adopted in the inner iterations we find the Conjugate Gradient (CG) method. In the convex case (positive definitive Hessian), it performs stably, but in the case of indefinite Hessian it may untimely breakdown (pivot breakdown) or become numerically unstable. This may occur when in the CG iterations a direction $s$ such that $s^\top \nabla^2 f(x_k)s < 0$ is encountered before satisfying the termination criterion. As well known, such directions are called *negative curvature directions* for the function $f$ at $x_k$ and, as we will discuss afterwards, they may play an important rule for improving both the converge properties and the efficiency of the method. To overcome the drawback of the CG untimely stopping, some authors proposed the use of the Lanczos process in the inner iterations in place of the CG [30, 34]. The two methods are, to some extent, equivalent in the case of convex functions, but the iterations of the Lanczos process do not prematurely stop in the case of indefinite Hessian.

Negative curvature directions may be fruitfully exploited within a Truncated Newton method for improving its convergence properties and performance. Their first use dates back to the seminal papers [31, 32] and several methods have been then developed, based on a combination of a Newton–type direction $d_k$ and a negative curvature

direction $s_k$. In such methods the iterative scheme (1.2) is replaced by

$$x_{k+1} = x_k + \alpha_k^2 d_k + \alpha_k s_k, \tag{1.4}$$

and $\alpha_k$ is obtained by means of a *curvilinear linesearch*. The use of negative curvature directions has a twofold importance: from the computational point of view, a movement along a descent negative curvature direction allows the algorithm to escape from regions of nonconvexity of the objective function. From the theoretical point of view, the use of appropriate negative curvature directions enables defining methods converging to second order critical points, i.e., points which satisfy second order necessary optimality conditions (stationary points where the Hessian is positive semidefinite). Newton–type methods based on trust region approach naturally possess such convergence property (see [11]), whilst linesearch–based methods require the following strong assumption on used the negative curvature direction: it must be an approximation of an eigenvector of the Hessian matrix corresponding to its most negative eigenvalue. More precisely, to guarantee second order convergence, the negative curvature direction $s_k$ is basically required to be a *bounded descent direction* satisfying the following property:

$$s_k^\top \nabla^2 f(x_k) s_k \longrightarrow 0 \quad \text{implies} \quad \min\left\{0, \lambda_{\min}\left[\nabla^2 f(x_k)\right]\right\} \longrightarrow 0, \tag{1.5}$$

where $\lambda_{\min}\left[\nabla^2 f(x_k)\right]$ is the smallest eigenvalue of the Hessian matrix $\nabla^2 f(x_k)$. Computing a direction $s_k$ satisfying (1.5) is a very computationally expensive task, since it involves the spectrum of $\nabla^2 f(x_k)$. Moreover, most of the strategies proposed in literature for computing negative curvature directions satisfying (1.5) usually rely on matrix factorizations (see e.g., the Bunch and Parlett decomposition proposed in [25, 32]), so that in the large scale setting they become impracticable. On the other hand, also iterative methods usually adopted typically need to store a large matrix, hence they are unsuited for handling large scale problems; this is the case of the method proposed in [30] where (in the framework of nonmonotone methods) the Lanczos process is used and the storage of a matrix of the Lanczos vectors generated at each iteration is theoretically needed to compute adequate negative curvature directions. A different approach for computing suited negative curvature directions is proposed in [26]. In this paper, based on the close relation between CG and Lanczos methods, the Lanczos vectors are regenerated by rerunning the recurrence when needed. In this manner, matrix storage is avoided, but a non-negligible additional computational effort is required, due to rerunning operations.

To these authors' knowledge, a first attempt, in the case of indefinite Hessian, to iteratively compute negative curvature directions satisfying (1.5), without requiring storage of any large matrix or rerunning the iterative process, is represented by the use of the Planar-CG algorithm as proposed in [24] (we refer the reader to the papers [19, 20] for a complete description of the Planar-CG schemes). Besides providing a general theoretical framework which guarantees convergence to second order critical points, results of a preliminary numerical experience reported in [24] show that the proposed approach is reliable and promising. Nevertheless, we believe that there is still need to further investigate on how to determining effective negative curvature directions

to be used within a Truncated Newton method. In particular, besides guaranteeing convergence toward second order critical points, the use of such directions should improve the overall efficiency of the method and its capability to detect *better local minimizers*.

Another issue worth investigating concerns how to combine a Newton–type direction and a negative curvature direction taking into account their possible different scaling. As well known, Newton–type direction is well–scaled (particularly when close to a local minimizer), while a negative curvature direction may be possibly not. Hence, inefficiency may arise due to the use of a combination of the two directions. Based on this remark, in [26], at each outer iteration, given a descent pair of directions $(d_k, s_k)$, instead of using the iterative scheme (1.4), only one of the two directions is selected and a suited linesearch is performed along the chosen direction. The selection of the most promising direction is performed by estimating the rate of decrease of the quadratic model of the objective function in both directions. Following this approach, in [21], a new Truncated Newton method is proposed; a test based on the quadratic model is used to select the most promising between the two directions and an appropriate linesearch procedure is adopted depending on the selected search direction. On the same guideline of selecting the most effective direction, in [35] the authors propose to consider three alternatives: to select one of the two directions $d_k$ and $s_k$, or possibly to make use of a combination of both the directions. This latter choice is adopted when both the directions are promising in terms of decrease of the quadratic model. On the other hand, as studied in [1], it would be very beneficial to perform a scaling process before combining the two directions. We also mention that in the recent paper [13], a novel framework has been proposed for combining the two directions, alternating two–step and dynamic step approaches.

It is important to point out that, even if Truncated Newton methods use second order information on the objective function, actually they are "matrix–free" (Hessian–free), since Hessian matrix is never stored and it is accessed by means of a routine which provides the matrix–vector product of the Hessian times a vector. This feature, along with the scale invariance property, makes these methods very attractive also in recently raised machine learning applications (including deep neural networks). Hessian times vector products can be computed at a cost which is a small multiple of the cost of a gradient evaluation (see, e.g., [3]). Such problems arise, for instance, in training deep neural networks, in low rank subspace clustering problems [29] and in many other applications. Moreover, in several contexts (see, e.g., statistical physics, random matrix theory and training multilayer perceptron neural networks [2, 4, 10, 14]) negative curvature directions are a useful tool for an algorithm to escape from saddle points that may represent a frequently arising obstacle. As clearly pointed out in the recent paper by Curtis and Robinson [13], today it is really needful to design new methods able to efficiently solve nonconvex problems by exploiting negative curvature directions, both in deterministic and stochastic optimization. These authors also affirm that few algorithms converging to second order critical points have been up to now developed. They indicate that the main reasons for this rely on an excessive computational burden due to the computation of appropriate negative curvature directions, along with a not sufficient evidence of the full benefit of incorporating such directions.

Finally observe that negative curvature directions, obtained as by–product of iterative optimization methods, have also been investigated within the recent literature related to preconditioners for large scale linear systems. In particular, quasi–Newton based updates for the construction of preconditioners were proposed, both within Truncated Newton methods and Nonlinear CG methods, where the combined use of both positive and negative curvature directions can be fruitfully exploited (see, e.g., [7] and [8]).

In this paper, we propose the use of an alternative iterative procedure to be used within Newton–Krylov methods, for computing an effective negative curvature direction. In particular, we refer to SYMMBK method for solving large scale symmetric possibly indefinite linear systems [5, 9, 11, 28]. Such method has been recently successfully applied within Truncated Newton methods to yield a gradient related Newton–type direction [6]. More precisely, in the last paper a modified Bunch–Kaufmann factorization has been proposed within SYMMBK algorithm for solving the Newton equation, at each outer iteration. Indeed, the Bunch–Kaufmann factorization is an effective and stable matrix decomposition, but when used for solving the Newton equation might provide a direction which is not gradient–related. The modification proposed in [6] enables obtaining a direction that is gradient–related and effective in practice. Hence, the idea in the current paper to possibly use the same procedure based on the modified SYMMBK algorithm for obtaining also a negative curvature direction satisfying (1.5), with a minimal additional storage. The latter storage requirement is reduced to just one additional vector, by means of replacing the use of the CG method. We prove some theoretical achievements associated with the novel negative curvature direction we adopt. Then we report the results of an extensive numerical experimentation showing the reliability and the effectiveness of the proposed approach. Such results have been analyzed also through a novel profiling tool, the *Quality Profiles*, which highlight the good capability of the algorithms incorporating negative curvature directions to determine better local minimizers.

The paper is organized as follows. In Section 2 we briefly recall the SYMMBK procedure and provide some preliminaries on the use of negative curvature directions within a Truncated Newton method. Sections 3 describe how to use SYMMBK for defining a suited negative curvature direction. Section 4 deals with the iterative computation of negative curvature directions. Theoretical results concerning the computed negative curvature directions are included in Section 5. The results of the numerical experimentation are reported in Section 6. Finally, Section 7 includes some concluding remarks. As regards the notations, given a set $I$, $|I|$ denotes its cardinality; vectors $\{e_j\}$ indicate the canonical basis of $\mathbb{R}^n$ and $\|v\|$ specifies the Euclidean norm of $v \in \mathbb{R}^n$; given a square matrix $A$, $\lambda_{\min}[A]$ denotes its smallest eigenvalue and $\kappa(A)$ its Euclidean condition number.

## 2 Preliminaries

SYMMBK method was originally proposed in [9]. It iteratively solves a symmetric linear system basically relying on the following two relevant tools: *the Lanczos iterative process* for the reduction of the original system to a symmetric tridiagonal system;

*the Bunch–Kaufmann decomposition* of tridiagonal matrices, through an appropriate pivoting strategy. Let us consider system (1.3), namely the Newton equation at the $k$-th outer iteration of the Truncated Newton method, where the matrix $\nabla^2 f(x_k)$ is possibly indefinite. The first tool allows to transform the symmetric linear system (1.3) into the system

$$\begin{cases} T_k y_k = \|\nabla f(x_k)\| e_1 \\ d_k = Q_k y_k, \end{cases} \tag{2.1}$$

being $T_k \in \mathbb{R}^{m\times m}$ symmetric and *tridiagonal*, and $Q_k \in \mathbb{R}^{n\times m}$, where $m$ is the number of iterations performed by the Lanczos process. The columns of the matrix $Q_k$ are given by the $m$ Lanczos vectors (see also [12])

$$Q_k = \begin{pmatrix} q_1 & \vdots & \vdots & q_m \end{pmatrix}, \tag{2.2}$$

with $q_\ell^\top q_i = 0$ and $\|q_\ell\| = 1$, being $1 \le \ell \ne i \le m$. We recall that, unlike the CG method, on indefinite symmetric linear systems the Lanczos process does not suffer for a possible pivot breakdown. A relevant property of matrix $Q_k$ is that it results

$$T_k = Q_k^\top \nabla^2 f(x_k) Q_k. \tag{2.3}$$

The Bunch–Kaufmann decomposition in SYMMBK allows for an easy factorization of the tridiagonal matrix $T_k$ as in

$$T_k = S_k B_k S_k^\top, \tag{2.4}$$

being $S_k \in \mathbb{R}^{m\times m}$ a *block unit lower triangular* matrix, while the matrix $B_k \in \mathbb{R}^{m\times m}$ is *block diagonal*, with blocks of possible dimensions $1\times 1$ or $2\times 2$. By (2.1), after $m$ iterations of the Lanczos process, the vector $d_k$ represents an approximate solution of (2.1) and it can be used as a search direction within an optimization framework. Furthermore, in [6] the authors slightly modified the pivoting rule within the Bunch–Kaufmann decomposition, so that the resulting vector $d_k$ is provably a *gradient–related* direction for the optimization framework where SYMMBK is used.

Our main task here is represented by exploiting SYMMBK procedure, in order to iteratively build an effective negative curvature direction $s_k \in \mathbb{R}^n$ for $f(x)$ at $x_k$, so that no more than one $n$–dimensional vector needs to be stored for its computation. This will provide a general matrix–free technique to construct negative curvature directions in large scale settings, where a popular and well renowned tool, namely SYMMBK procedure, is adopted. The vector $s_k$ will be used within a Truncated Newton method to solve (1.1), in order to steer the convergence towards a stationary point where the Hessian matrix is positive semidefinite. As a more specific task, we hereafter technically address the negative curvature direction $s_k$ such that, at each outer iteration $k$, the following conditions are fulfilled:

$$\begin{aligned} &\text{(i)} \quad s_k^\top \nabla f(x_k) < 0, \text{ for any } s_k \ne 0; \\ &\text{(ii)} \quad s_k^\top \nabla^2 f(x_k) s_k < 0, \text{ for any } s_k \ne 0; \\ &\text{(iii)} \quad s_k^\top \nabla^2 f(x_k) s_k \to 0 \quad \Longrightarrow \quad \min\{0, \lambda_{\min}[\nabla^2 f(x_k)]\} \to 0. \end{aligned} \tag{2.5}$$

Observe that (i) in (2.5) merely imposes that $s_k$ (if any) is a descent direction for $f(x)$ at $x_k$, while (ii) in (2.5) claims that $s_k$ has a nonzero projection on eigenvectors of $\nabla^2 f(x_k)$ associated with negative eigenvalues. Finally, (iii) in (2.5) imposes that, broadly speaking, when we approach a region of convexity for $f(x)$, then $s_k$ eventually approaches a vector in the null space of $\nabla^2 f(x_k)$: this condition guarantees convergence to second order critical points. Conditions (i)–(iii) in (2.5) allow, in our algorithmic framework, to compute the next iterate $x_{k+1}$ according with (1.4), using a modified Armijo–type curvilinear linesearch procedure (see, e.g., [31]).

# 3 On computing the negative curvature direction $s_k$

Let the vector $w \in \mathbb{R}^m$ be an eigenvector of the matrix $B_k$ given in (2.4), associated with a negative eigenvalue $\lambda$. Furthermore, assume that computing the vector $y \in \mathbb{R}^m$ such that $S_k^\top y = w$ represents a relatively simple task. Then, by (2.3) and (2.4) we obtain

$$\begin{aligned}(Q_k y)^\top \nabla^2 f(x_k)(Q_k y) \;&=\; y^\top \left[Q_k^\top \nabla^2 f(x_k) Q_k\right] y \;=\; y^\top T_k y \;=\; y^\top S_k B_k S_k^\top y \\ &=\; (S_k^\top y)^\top B_k (S_k^\top y) \;=\; w^\top B_k w \;=\; \lambda \|w\|^2 < 0.\end{aligned}$$

Thus, the vector $Q_k y$ represents a negative curvature direction for the function $f(x)$ at $x_k$, and in the sequel we are committed to yield a reliable procedure such that the subsequent results hold:

- the efficient (say iterative) computation of the vector $s_k = Q_k y$, exploiting SYMMBK procedure, without storing any matrix;
- the fulfillment of the conditions (i)–(iii) in (2.5) for $s_k$.

On this purpose we preliminary consider the next result, whose proof can be easily obtained from Lemma 4.3 in [32] and Theorem 3.2 in [24].

**Lemma 3.1.** *Let us consider the problem (1.1), along with the sequence $\{x_k\}$. Suppose $m = n$ iterations of the Lanczos process are performed by SYMMBK when solving Newton's equation (1.3) at iterate $x_k$, for a given $k \geq 1$, so that the decompositions*

$$\begin{cases} T_k = Q_k^\top \nabla^2 f(x_k) Q_k \\ T_k = S_k B_k S_k^\top \end{cases} \tag{3.1}$$

*are available. Then, $Q_k \in \mathbb{R}^{n\times n}$ is orthogonal and $T_k \in \mathbb{R}^{n\times n}$ has the same eigenvalues of $\nabla^2 f(x_k)$; moreover, the matrices $S_k \in \mathbb{R}^{n\times n}$ and $B_k \in \mathbb{R}^{n\times n}$ are nonsingular. In addition, if $w$ is a unit eigenvector corresponding to the smallest negative eigenvalue $\lambda$ of $B_k$, and $\bar{y}$ is a (bounded) solution of the linear system $S_k^\top y = w$, then the vector $s_k = Q_k \bar{y}$ is a bounded direction that satisfies (i)–(iii) in (2.5).*

Now, observe that the results in Lemma 3.1 assume that the Lanczos process performs exactly $n$ iterations to solve (1.3): this is definitely unaffordable for large $n$. Hence, we need to generalize the contents in Lemma 3.1 to the case $m < n$. Moreover, we highlight that to compute the vector $\bar{y}$ in Lemma 3.1 we can resort to Lemma 4.3

in [32]. In this regard, with the next lemma we intend to rephrase Lemma 3.1, though obtaining weaker conclusions, being possibly (iii) in (2.5) not fulfilled.

**Lemma 3.2.** *Let us consider the problem (1.1), along with the sequence* $\{x_k\}$. *Suppose* $m < n$ *iterations of the Lanczos process are performed by SYMMBK when solving Newton's equation (1.3) at iterate* $x_k$, *for a given* $k \geq 1$, *so that the decompositions (3.1) are available. Then, we have* $Q_k \in \mathbb{R}^{n\times m}$ *and* $T_k \in \mathbb{R}^{m\times m}$, *along with the fact that the matrices* $S_k \in \mathbb{R}^{m\times m}$ *and* $B_k \in \mathbb{R}^{m\times m}$ *are nonsingular. In addition, if* $w$ *is a unit eigenvector corresponding to the smallest negative eigenvalue* $\lambda$ *of* $B_k$, *and* $\bar{y}$ *is a (bounded) solution of the linear system* $S_k^\top y = w$, *then the vector* $s_k = Q_k\bar{y}$ *is a bounded direction that satisfies (i)–(ii) in (2.5).*

As a further result, Lemma 4.3 in [32] ensures that the outcomes in Lemma 3.1 can be easily generalized when the linear system $S_k^\top y = w$ is replaced by

$$S_k^\top y = \sum_{\substack{1\leq i\leq m\\ \lambda_i<0}} w_i, \tag{3.2}$$

being $w_i$ the eigenvector of $B_k$ corresponding to a negative eigenvalue $\lambda_i$. In this regard, some additional observations require our attention:

- computing all the unit eigenvectors of the matrix $B_k$ may represent in general an expensive task, so that we may limit our analysis to compute an eigenvector associated to (one of) the smallest eigenvalues of $B_k$, then exploiting Lemma 3.1;
- fully computing all the eigenvectors of $B_k$ does not ensure that an undoubtedly more effective negative curvature direction $s_k$ will be available;
- the computation of the smallest negative eigenvalue of the matrix $B_k$ is considerably simplified by exploiting a diagonalization of $B_k$.

Now we focus on the last issue, namely a suitable diagonalization of the block diagonal matrix $B_k$ in (3.1), in order to simplify the computation of its eigenpairs. To this aim, let $D_k \in \mathbb{R}^{m\times m}$ be a diagonal matrix, with $D_k = diag\{\lambda_1, \ldots, \lambda_m\}$ and where $\lambda_1, \ldots, \lambda_m$ are all the eigenvalues (possibly not all distinct) of $B_k$, and let $X_k \in \mathbb{R}^{m\times m}$ be an orthogonal matrix, such that its columns correspond to the eigenvectors of $B_k$ associated to the eigenvalues $\lambda_1, \ldots, \lambda_m$. Then we have $D_k = X_k^\top B_k X_k$, i.e.,

$$B_k = X_k D_k X_k^\top. \tag{3.3}$$

Hence, since $B_k$ is a block diagonal matrix (with $1\times 1$ and $2\times 2$ blocks), then also $X_k$ will be a block diagonal matrix with blocks of dimension at most $2\times 2$. In particular, for any $1\times 1$ diagonal block $B^{(i\ i)}$ [$2\times 2$ diagonal block $B^{(i\ i+1)}$] of matrix $B_k$, we will have the corresponding $1\times 1$ block $X^{(i\ i)}$ [$2\times 2$ block $X^{(i\ i+1)}$] of matrix $X_k$, corresponding to the number 1 [to the two eigenvectors of the sub–matrix $B^{(i\ i+1)}$]. As an example, considering the case $B_k$ given by all $1\times 1$ diagonal blocks apart from

the block $B^{(i\ i+1)}$; then we have:

$$B_k = \begin{pmatrix} * & & & & \\ & * & & & \\ & & B^{(i\ i+1)} & & \\ & & & * & \\ & & & & * \end{pmatrix}, \qquad X_k = \begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & X^{(i\ i+1)} & & \\ & & & 1 & \\ & & & & 1 \end{pmatrix}, \tag{3.4}$$

where $X^{(i\ i+1)} = (v^i \vdots v^{i+1}) \in \mathbb{R}^{2\times 2}$ and $v^i, v^{i+1}$ are the unit eigenvectors of the $2 \times 2$ block $B^{(i\ i+1)}$. This last example shows that the computation of the eigenpairs of $B_k$ is relatively easy, since it is a block diagonal matrix with at most $2 \times 2$ blocks.

Hereafter we will indicate with $\lambda_\ell = \lambda_{\min}[B_k]$ the smallest (negative) eigenvalue of $B_k$, and $z_\ell$ will be its corresponding unit eigenvector. Thus, $\ell$ will be used to denote the row (column) index corresponding to the smallest (negative) diagonal entry of matrix $D_k$. By (2.4) and (3.3) we immediately have

$$T_k = S_k X_k D_k X_k^\top S_k^\top = W_k D_k W_k^\top, \qquad \text{where} \quad W_k \stackrel{\text{def}}{=} S_k X_k \in \mathbb{R}^{m\times m}. \tag{3.5}$$

Now, in order to exploit the theory indicated in Lemma 3.1 and Lemma 3.2, we need to compute the eigenvector corresponding to the smallest eigenvalue of $T_k$, so that a negative curvature direction for $f(x)$ at $x_k$ can be computed, fulfilling (i)–(ii) in (2.5) and possibly (iii). On this purpose, we have to do nothing else but replacing in (3.2) the matrix $S_k$ with the matrix $W_k$ in (3.5). Hence, on the overall the computation of the negative curvature direction $s_k$ requires solving the linear system

$$W_k^\top y = z_\ell. \tag{3.6}$$

By simple inspection of (3.6), after exploiting the block structure of the matrix $W_k$ we can realize that, following the guidelines in [15] and [24], a straightforward backtracking algorithm allows the computation of the solution $\bar{y} \in \mathbb{R}^m$ for (3.6). However, after some computations, (see [22]) one realizes that, following this approach, the computation of $s_k$ requires the storage of the vectors $q_1, \ldots, q_m$, inasmuch as the index $\ell$ will be known only at the end of the $m$-th Lanczos iteration. This makes the iterative backtracking procedure to solve (3.6) impracticable for large scale problems, justifying an alternative method proposed in the next section.

# 4 A better exploitation of SYMMBK to compute negative curvature directions

In this section, following the approach adopted in [11, Section 5.2] for finding conjugate directions from an orthonormal Krylov basis, we propose to better exploit SYMMBK to generate $\nabla^2 f(x_k)$–conjugate vectors to be used for computing negative curvature directions.

Observe that the matrix $W_k$ in (3.5) has the form

$$W_k = S_k X_k = \begin{pmatrix} W^{(1\ 1)} & & & & \\ W^{(2\ 1)} & W^{(2\ 2)} & & & \\ & \cdot & \cdot & & \\ & & \cdot & W^{(j-1\ j-1)} & \\ & & & W^{(j\ j-1)} & W^{(j\ j)} \end{pmatrix}, \qquad j \geq 1, \tag{4.1}$$

where the sizes (both rows and columns) of the sub–diagonal blocks $W^{(i+1\ i)}$, $i = 1, \ldots, j-1$, depend on the sizes of $W^{(i\ i)}$ and $W^{(i+1\ i+1)}$ diagonal blocks[1]. Moreover, recalling that $S_k$ is block unit lower triangular, then the diagonal blocks $W^{(i\ i)} \equiv X^{(i\ i+1)}$ are orthogonal (see also (3.4)). Now, by combining (2.4), (3.1) and (3.5) we can compute a set of $\nabla^2 f(x_k)$–conjugate directions, being indeed

$$T_k = W_k D_k W_k^\top = Q_k^\top \nabla^2 f(x_k) Q_k, \tag{4.2}$$

so that

$$D_k = W_k^{-1} Q_k^\top \nabla^2 f(x_k) Q_k W_k^{-T} = G_k^\top \nabla^2 f(x_k) G_k, \tag{4.3}$$

where

$$G_k \stackrel{\text{def}}{=} Q_k W_k^{-T} \in \mathbb{R}^{n \times m}. \tag{4.4}$$

Since $D_k$ is a diagonal matrix, by (4.3)–(4.4) the columns of $G_k$ yield a set of $m$ linearly independent (see also Proposition 2.1 of [19] and [20]) $\nabla^2 f(x_k)$–conjugate directions which span the Krylov subspace $\mathcal{K}(\nabla^2 f(x_k), q_1, m)$. To efficiently compute the matrix $G_k$, let us define

$$G_k = (G^1\ G^2\ \ldots\ G^{j-1}\ G^j), \tag{4.5}$$

being $G^i$ an $n \times 1$ or an $n \times 2$ sub–matrix, for any $1 \leq i \leq j$. Thus, we can now re–write equation (4.4) as

$$G_k W_k^T = Q_k \stackrel{\text{def}}{=} (Q^1\ Q^2\ \ldots\ Q^{j-1}\ Q^j), \tag{4.6}$$

where each $Q^i$, $1 \leq i \leq j$, represents an $n \times 1$ or an $n \times 2$ matrix whose columns are given by Lanczos vectors, in accordance with the structure of $G_k$ in (4.5). Hence, using the expression (4.1) for matrix $W_k$, as well as the orthogonality of the blocks $W^{(i\ i)}$, $1 \leq i \leq j$, we obtain from (4.5) and (4.6)

$$G^i = \left[ Q^i - G^{i-1} \left( W^{(i\ i-1)} \right)^\top \right] W^{(i\ i)}, \qquad 1 < i \leq j, \tag{4.7}$$

with $G^1 = Q^1 W^{(1\ 1)}$. Hence, we can efficiently and iteratively compute the blocks $\{G^i\}$, whose columns represent $\nabla^2 f(x_k)$–conjugate directions, as long as the quantities $\{Q^j\}$, $\{W^{(i\ i-1)}\}$ and $\{W^{(i\ i)}\}$ are available.

[1] E.g., in case $W^{(2\ 2)} \in \mathbb{R}^{1 \times 1}$ and $W^{(3\ 3)} \in \mathbb{R}^{2 \times 2}$, then we will have $W^{(3\ 2)} \in \mathbb{R}^{2 \times 1}$.

# 5 Theoretical results for negative curvature directions computation

Relations (4.7) indicate how to fully iteratively compute the matrix $G_k$ in (4.4). Moreover, (4.3) indicates that the columns of $G_k$ represent indeed a set of $\nabla^2 f(x_k)$–conjugate vectors; we denote by $G_{(j)}$, $1 \leq j \leq m$, such columns[2]. Let us define the index set $\mathcal{J} = \{j \in \{1, ..., m\} \mid \mu_j < 0\}$, where $\mu_j$ is the $j$-th eigenvalue of the diagonal matrix $D_k$ in (3.3). Now we can define the vector

$$z = \sum_{j \in \mathcal{J}} a_j G_{(j)}, \tag{5.1}$$

where the coefficients $a_j \in \mathbb{R}$, $a_j \neq 0$, $j = 1, \ldots, m$, are such that

$$\sum_{j \in \mathcal{J}} a_j^2 \mu_j \leq \lambda_{\min}\left[D_k\right] \min_{j \in \mathcal{J}} a_j^2. \tag{5.2}$$

Our aim is to show that the vector $z$ in (5.1) can be used as negative curvature direction of the function $f(x)$ at $x_k$. In particular, in the theoretical results which follow, in order to prove that our final negative curvature direction fulfills (i)–(iii) in (2.5), we consider the normalized direction

$$s_k = z/\|z\|. \tag{5.3}$$

Note that the use of the direction $s_k$, in place of $z$, is only for theoretical purpose; a similar approach is used in [32], where, in the theoretical analysis, both the search directions in (1.4) are assumed to be bounded.

**Proposition 5.1.** *Given the function $f : \mathbb{R}^n \to \mathbb{R}$, with $f \in C^2(\mathbb{R}^n)$, let us consider the sequence $\{x_k\}$ of approximate solutions to problem (1.1). Assume at iterate $x_k$ the Hessian matrix $\nabla^2 f(x_k)$ has at least one negative eigenvalue. Let $G_k \in \mathbb{R}^{n \times n}$ be the matrix in (4.4) after $n$ inner iterations, and let $\kappa(G_k)$ denote the condition number of $G_k$. Assume $\{a_j\}$ (with $1 \leq j \leq n$) is a set of real values satisfying (5.2), and $\{\mu_j\}$ is the set of eigenvalues of the diagonal matrix $D_k$. Then*

$$s_k^\top \nabla^2 f(x_k) s_k \;\leq\; \frac{1}{N \cdot [\kappa(G_k)]^2} \, \frac{\min\limits_{j \in \mathcal{J}} a_j^2}{\max\limits_{j \in \mathcal{J}} a_j^2} \, \lambda_{\min}\left[\nabla^2 f(x_k)\right], \tag{5.4}$$

*where $N \geq 1$ is the number of negative eigenvalues of $\nabla^2 f(x_k)$.*

*Proof.* Let us consider the unit eigenvector $u_{\min}$ of $\nabla^2 f(x_k)$ corresponding to the smallest eigenvalue $\lambda_{\min}\left[\nabla^2 f(x_k)\right]$. Hence, $u_{\min}^\top \nabla^2 f(x_k) u_{\min} = \lambda_{\min}\left[\nabla^2 f(x_k)\right]$ and therefore by (4.3)

$$\lambda_{\min}\left[\nabla^2 f(x_k)\right] = u_{\min}^\top G_k^{-T} D_k G_k^{-1} u_{\min} = w^\top D_k w \geq \lambda_{\min}\left[D_k\right] \|w\|^2,$$

[2] Again, for simplicity, we drop the dependency of $G_{(j)}$ on the index $k$.

where $w = G_k^{-1} u_{\min}$. Now, since $\|w\| \leq \left\|G_k^{-1}\right\| \|u_{\min}\| = \left\|G_k^{-1}\right\|$ we have

$$\lambda_{\min}\left[\nabla^2 f(x_k)\right] \geq \lambda_{\min}\left[D_k\right] \|w\|^2 \geq \lambda_{\min}\left[D_k\right] \left\|G_k^{-1}\right\|^2. \tag{5.5}$$

From (5.1) we have

$$z \;=\; G_k \sum_{j \in \mathcal{J}} a_j e_j \tag{5.6}$$

so that, from (5.2) we also obtain

$$\begin{aligned} z^\top \nabla^2 f(x_k) z \;=\; z^\top G_k^{-T} D_k G_k^{-1} z \;&=\; \left[\sum_{j \in \mathcal{J}} a_j e_j\right]^\top D_k \left[\sum_{j \in \mathcal{J}} a_j e_j\right] \\ &= \sum_{j \in \mathcal{J}} a_j^2 \mu_j \;\leq\; \lambda_{\min}\left[D_k\right] \min_{j \in \mathcal{J}} a_j^2. \end{aligned}$$

Hence, by (5.5) it follows that

$$z^\top \nabla^2 f(x_k) z \left\|G_k^{-1}\right\|^2 \leq \lambda_{\min}\left[D_k\right] \min_{j \in \mathcal{J}} a_j^2 \;\; \left\|G_k^{-1}\right\|^2 \leq \min_{j \in \mathcal{J}} a_j^2 \; \lambda_{\min}\left[\nabla^2 f(x_k)\right] \;<\; 0.$$

Moreover, since

$$\|z\|^2 \leq \|G_k\|^2 \left\|\sum_{j \in \mathcal{J}} a_j e_j\right\|^2 \leq N \|G_k\|^2 \max_{j \in \mathcal{J}} a_j^2,$$

then by (5.6)

$$\begin{aligned} 0 \;>\; \lambda_{\min}\left[\nabla^2 f(x_k)\right] \;&\geq\; \frac{z^\top \nabla^2 f(x_k) z \|G_k^{-1}\|^2}{\min\limits_{j \in \mathcal{J}} a_j^2} \\ &\geq\; N \|G_k^{-1}\|^2 \|G_k\|^2 \frac{\max\limits_{j \in \mathcal{J}} a_j^2}{\min\limits_{j \in \mathcal{J}} a_j^2} \frac{z^\top \nabla^2 f(x_k) z}{\|z\|^2}, \end{aligned} \tag{5.7}$$

so that, recalling (5.3), condition (5.4) holds. □

The last proposition evidences that in case the Lanczos process is able to perform exactly $n$ inner iterations within SYMMBK procedure, then a unit negative curvature direction $s_k$ satisfying (i)–(iii) in (2.5) can be easily available as by–product from (4.4), (5.1) and (5.3). We also remark that the computation of the vector $z$ in (5.1) does not require the storage of more than one additional vector, i.e. the current sum of the contributions $\{a_j G_{(j)}\}$ up to the $m$-th inner iteration. The latter result, to these authors' knowledge, represents the first example in the literature of a so cheap computation of the vector $s_k$ fulfilling (i)–(iii) in (2.5).

**Remark 5.2.** Relation (5.4) reveals that the effectiveness of the negative curvature direction $s_k$ requires the boundedness of $\kappa(G_k)$. Nevertheless, in case the quantity $\|G_k^{-1}\|$ is itself bounded, by (5.7) we can alternatively conclude that also the vector $z$ (and not only the vector $s_k$) satisfies (iii) in (2.5). Thus, $z$ could be used as an alternative negative curvature direction, too, in place of $s_k$. In this regard, by relations (2.18) and (3.15) in [6] (where the matrix $W_k$ plays the role of the matrix $G_k$ in the current paper), and recalling that $S_k = -S_k^{-1}$, the proper choice of the parameter $\omega$ in [6] can ensure that the quantity $\|G_k^{-1}\|$ is indeed bounded. This partially fills the gap between the current paper and [6], where a proper choice of the parameter $\omega$ was needed in order to compute a gradient–related direction by SYMMBK. In this regard, those values of $\omega$ selected in [6] are also worth in the current paper for computing an effective negative curvature direction (see also Section 5.1 for additional considerations).

Observe that the fulfillment of condition (5.2) is a preliminary requirement for the construction of the negative curvature direction $s_k$ to be used within Proposition 5.1. Hence, in the next result we show how to iteratively properly select the coefficients $\{a_j\}$ in (5.1) so that (5.2) holds.

**Lemma 5.3.** *Let us consider the sequence of the real coefficients $\{a_j\}$ in (5.1) and (5.2). For any $h = 1, \ldots, m$, let us define the real quantities:*

$$\lambda_{\min}^{(h-1)} = \min_{\substack{1 \le j \le h-1 \\ \mu_j < 0}} \{\mu_j\}, \qquad C^{(h-1)} = \min_{\substack{1 \le j \le h-1 \\ \mu_j < 0}} \{a_j^2\}, \qquad A^{(h-1)} = \sum_{\substack{1 \le j \le h-1 \\ \mu_j < 0}} a_j^2 \mu_j. \tag{5.8}$$

*Condition (5.2) is fulfilled provided that for any $j \ge 2$, when $\mu_j > \lambda_{\min}^{(j-1)}$ then we set*

$$a_j^2 \le \min \left\{ C^{(j-1)}, \frac{-A^{(j-1)}}{\mu_j - \lambda_{\min}^{(j-1)}} \right\}, \qquad j = 1, \ldots, m. \tag{5.9}$$

*Proof.* The proof proceeds by induction. Relation (5.2) clearly holds when $j = 1$, with no specific assumption on $a_1$. Then, we assume that it holds for $j-1$, and we prove the result for the index $j$. Observe that for any $j$ in (5.1)-(5.2) we have $\mu_j < 0$; moreover, the condition

$$A^{(j-1)} \le \lambda_{\min}^{(j-1)} C^{(j-1)} \tag{5.10}$$

is satisfied by inductive hypothesis. Therefore, for the index $j$ we analyze the conditions which guarantee that the inequality (5.2), i.e.

$$A^{(j-1)} + a_j^2 \mu_j \le \min \left\{ \lambda_{\min}^{(j-1)} \, , \, \mu_j \right\} \min \left\{ C^{(j-1)}, a_j^2 \right\} \tag{5.11}$$

is satisfied. This yields the next two cases:

– if $\mu_j < \lambda_{\min}^{(j-1)}$, then (5.11) yields $A^{(j-1)} + a_j^2\mu_j \leq \mu_j \min\left\{C^{(j-1)}, a_j^2\right\}$, so that in case $C^{(j-1)} < a_j^2$, since $C^{(j-1)} \geq 0$, we obtain

$$A^{(j-1)} + a_j^2\mu_j \leq \mu_j C^{(j-1)} \leq \lambda_{\min}^{(j-1)} C^{(j-1)}$$

which is always fulfilled recalling that $A^{(j-1)} + a_j^2\mu_j \leq A^{(j-1)}$ and considering relation (5.10). Conversely, in case $C^{(j-1)} \geq a_j^2$ we obtain

$$A^{(j-1)} + a_j^2\mu_j \leq \mu_j a_j^2$$

which is again always fulfilled inasmuch as $A^{(j-1)} < 0$;
– if $\mu_j \geq \lambda_{\min}^{(j-1)}$, then (5.11) yields $A^{(j-1)} + a_j^2\mu_j \leq \lambda_{\min}^{(j-1)} \min\left\{C^{(j-1)}, a_j^2\right\}$, where again we distinguish the case $C^{(j-1)} < a_j^2$, for which by (5.10) we have

$$A^{(j-1)} + a_j^2\mu_j \leq A^{(j-1)} \leq \lambda_{\min}^{(j-1)} C^{(j-1)},$$

that is always fulfilled, and the case $C^{(j-1)} \geq a_j^2$ which yields

$$\left(\mu_j - \lambda_{\min}^{(j-1)}\right) a_j^2 \leq -A^{(j-1)},$$

that holds by the condition (5.9) on the coefficient $a_j$.

□

We remark that the procedure to update the coefficients $\{a_j\}$ in Lemma 5.3 does not require the storage of any vector/matrix, so that on the overall the computation of the negative curvature direction $s_k$ can be iteratively carried on in large scale settings.

Unfortunately, the assumption in Proposition 5.1 that $n$ inner iterations (i.e., iterations of the Lanczos process) are performed is far from being realistic when $n$ is large, so that a possible extension of the results in Proposition 5.1 would be welcome for practical applications. In this regard, let us define

$$\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right] \stackrel{\text{def}}{=} \min_{\nu\in\mathbb{R}^m,\ \|\nu\|=1} \frac{\left[G_k\nu\right]^\top \nabla^2 f(x_k)\left[G_k\nu\right]}{\left\|G_k\nu\right\|^2}. \tag{5.12}$$

Observe that $\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right]$ represents the smallest value of the Rayleigh quotient for $\nabla^2 f(x_k)$, where the vector $G_k\nu$ spans the Krylov subspace $\mathcal{K}(\nabla^2 f(x_k), q_1, m)$. Hence, $\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right]$ can be regarded, to some extent, as an *approximation from above* (on the Krylov subspace $\mathcal{K}(\nabla^2 f(x_k), q_1, m)$) of $\lambda_{\min}\left[\nabla^2 f(x_k)\right]$, as stated in the next lemma.

**Lemma 5.4.** *Let us consider the matrix $G_k$ in (4.4) and the quantity $\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right]$ in (5.12). Then for $1 \leq h \leq n$ we have*

$$\lambda_{\min}\left[\nabla^2 f(x_k)\right] = \lambda_{\min}^{(n)}\left[\nabla^2 f(x_k)\right] \leq \cdots \leq \lambda_{\min}^{(h)}\left[\nabla^2 f(x_k)\right] \leq \cdots \leq \lambda_{\min}^{(1)}\left[\nabla^2 f(x_k)\right].$$

*Moreover, if the columns of the matrix $G_k = \left(G_{(1)} \cdots G_{(m)}\right) \in \mathbb{R}^{n\times m}$ in (5.12) are $\nabla^2 f(x_k)$–conjugate vectors, then*

$$\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right] = \min_{1\leq j\leq m} \frac{G_{(j)}^\top \nabla^2 f(x_k) G_{(j)}}{\left\|G_{(j)}\right\|^2}.$$

*Proof.* The result immediately follows from relation (5.12) and the inequality

$$\min_{\nu\in\mathbb{R}^h,\ \|\nu\|=1} \frac{\left[G_k\nu\right]^\top \nabla^2 f(x_k)\left[G_k\nu\right]}{\left\|G_k\nu\right\|^2} \geq \min_{\nu\in\mathbb{R}^m,\ \|\nu\|=1} \frac{\left[G_k\nu\right]^\top \nabla^2 f(x_k)\left[G_k\nu\right]}{\left\|G_k\nu\right\|^2},$$

for any $1 \leq h \leq m$. Moreover, by the conjugacy of the columns of $G_k$ we have

$$\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right] = \min_{\nu\in\mathbb{R}^m,\ \|\nu\|=1} \frac{\left[G_k\nu\right]^\top \nabla^2 f(x_k)\left[G_k\nu\right]}{\left\|G_k\nu\right\|^2} = \min_{1\leq j\leq m} \frac{G_{(j)}^\top \nabla^2 f(x_k) G_{(j)}}{\left\|G_{(j)}\right\|^2}$$

and this completes the proof. □

Using Lemma 5.4, we can give the next generalization of Proposition 5.1 to the case $m < n$ iterations of the Lanczos process are performed.

**Proposition 5.5.** *Given the function $f : \mathbb{R}^n \to \mathbb{R}$, with $f \in C^2(\mathbb{R}^n)$, let us consider the sequence $\{x_k\}$ of approximate solutions to problem (1.1). Assume at iterate $x_k$ the Hessian matrix $\nabla^2 f(x_k)$ has at least one negative eigenvalue. Let $G_k \in \mathbb{R}^{n\times m}$ be the matrix in (4.4) after $m < n$ inner iterations, and let $\sigma_{min}$ [$\sigma_{max}$] be its smallest [largest] singular value. Assume $\{a_j\}$, with $1 \leq j \leq n$, is a set of real values satisfying (5.2). Then*

$$s_k^\top \nabla^2 f(x_k) s_k \leq \frac{1}{m}\left(\frac{\sigma_{min}}{\sigma_{max}}\right)^2 \frac{\min\limits_{j\in\mathcal{J}} a_j^2}{\max\limits_{j\in\mathcal{J}} a_j^2}\, \lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right], \tag{5.13}$$

*where $\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right]$ is defined in (5.12).*

*Proof.* The proof follows the guidelines of Proposition 5.1, so that we will focus only on their differences. In particular we have from (4.3)

$$\nu^\top G_k^\top \nabla^2 f(x_k) G_k \nu = \nu^\top D_k \nu \geq \lambda_{\min}[D_k], \qquad \text{for all} \quad \nu \in \mathbb{R}^m,\ \|\nu\| = 1, \tag{5.14}$$

so that we can choose $\nu = \bar{\nu}$, with $\bar{\nu} \in \mathbb{R}^m$, $\|\bar{\nu}\| = 1$, in (5.14) such that (see also (5.12))

$$\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right] \;=\; \min_{\nu\in\mathbb{R}^m,\ \|\nu\|=1} \frac{\left[G_k\nu\right]^\top \nabla^2 f(x_k)\left[G_k\nu\right]}{\|G_k\nu\|^2} \;=\; \frac{\bar{\nu}^\top G_k^\top \nabla^2 f(x_k) G_k \bar{\nu}}{\|G_k\bar{\nu}\|^2}. \quad (5.15)$$

Now, we recall that by the *singular value decomposition* of $G_k = U\Sigma V^\top$, with $U \in \mathbb{R}^{n\times n}$, $V \in \mathbb{R}^{m\times m}$ and $\Sigma \in \mathbb{R}^{n\times m}$, there exist $m$ singular values[3] $\sigma_1, \ldots, \sigma_m$ such that

$$0 < \sigma_{\min} = \sigma_1 \leq \cdots \leq \sigma_j \leq \cdots \leq \sigma_m = \sigma_{\max}$$

and

$$G_k V = U \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_m \\ \hline & \emptyset_{n-m} & \end{bmatrix},$$

that is $G_k v_j = \sigma_j u_j$, $j = 1, \ldots, m$, being $m \leq n$, $U = (u_1 \cdots u_m u_{m+1} \cdots u_n)$ and $V = (v_1 \cdots v_m)$. Hence, $\|G_k v_j\| = \sigma_j \|u_j\|$, for $j = 1, \ldots, m$. Moreover, we have from this last relations, along with (4.3) and (5.15)

$$\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right] \;=\; \frac{\bar{\nu}^\top D_k \bar{\nu}}{\|G_k\bar{\nu}\|^2} \;\geq\; \frac{\lambda_{\min}[D_k]}{\|G_k\bar{\nu}\|^2} \;\geq\; \frac{\lambda_{\min}[D_k]}{\min\limits_{\|\nu\|=1}\|G_k\nu\|^2} = \frac{\lambda_{\min}[D_k]}{\sigma_{\min}^2}, \quad (5.16)$$

where the last inequality holds recalling that $\lambda_{\min}[D_k] < 0$. Furthermore, we have

$$z \;=\; \sum_{j\in\mathcal{J}} a_j G_{(j)} \;=\; G_k \sum_{j\in\mathcal{J}} a_j e_j,$$

and from (4.3) and (5.2) we have

$$\begin{aligned} z^\top \nabla^2 f(x_k) z \;&=\; \left[\sum_{j\in\mathcal{J}} a_j e_j\right]^\top G_k^\top \nabla^2 f(x_k) G_k \left[\sum_{j\in\mathcal{J}} a_j e_j\right] \;=\; \left[\sum_{j\in\mathcal{J}} a_j e_j\right]^\top D_k \left[\sum_{j\in\mathcal{J}} a_j e_j\right] \\ &=\; \sum_{j\in\mathcal{J}} a_j^2 \mu_j \;\leq\; \lambda_{\min}\left[D_k\right] \min_{j\in\mathcal{J}} a_j^2. \end{aligned}$$

Hence, from (5.16) and the last inequality we have

$$\frac{z^\top \nabla^2 f(x_k) z}{\sigma_{min}^2} \leq \frac{\lambda_{\min}\left[D_k\right] \min\limits_{j\in\mathcal{J}} a_j^2}{\sigma_{min}^2} \leq \lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right] \min_{j\in\mathcal{J}} a_j^2 < 0.$$

[3] Observe that since $G_k$ has rank $m$ then its $m$ singular values are positive.

Moreover, by (5.1) it results

$$\|z\|^2 \leq \|G_k\|^2 \left\| \sum_{j\in\mathcal{J}} a_j e_j \right\|^2 \leq \max_{\|\nu\|=1} \|G_k\nu\|^2 \sum_{j\in\mathcal{J}} a_j^2 \leq \sigma_{\max}^2 \sum_{j\in\mathcal{J}} a_j^2 \leq m\ \sigma_{\max}^2 \max_{j\in\mathcal{J}} a_j^2.$$

Thus, the last couple of relations yield

$$\frac{z^\top \nabla^2 f(x_k) z}{\|z\|^2} \leq \frac{\sigma_{\min}^2}{\|z\|^2} \min_{j\in\mathcal{J}} a_j^2\ \lambda_{\min}^{(k)} \left[\nabla^2 f(x_k)\right] \leq \frac{\sigma_{min}^2}{m\ \sigma_{max}^2} \frac{\min_{j\in\mathcal{J}} a_j^2}{\max_{j\in\mathcal{J}} a_j^2}\ \lambda_{\min}^{(k)} \left[\nabla^2 f(x_k)\right] < 0,$$

so that (5.13) holds. □

There is no difficulty to conclude that the contents in Lemma 5.3 and Remark 5.2 could be immediately extended to the results of Proposition 5.5, so that the iterative computation of a negative curvature direction can be fully exploited also when less than $n$ inner iterations are performed at the $k$-th outer iteration of the Truncated Newton method.

## 5.1 Issues on the choice of the sequence $\{a_j\}$

The presence of the sequence $\{a_j\}$ in Propositions 5.1 and 5.5 aims at giving generality in (5.1), when computing the negative curvature direction. In particular, on structured Hessian problems this may give an indication about those vectors $G_{(j)}$ to privilege for the computation of the negative curvature direction $s_k$. Conversely, the choice $a_j = 1$, for any $j \geq 1$, fulfills (5.2) and represents the most obvious one, since it also contributes to tighten the bounds in (5.4) and (5.13). Nevertheless, the choice $a_\ell = 1$, where $\ell = \arg\min_j \left\{ G_{(j)}^\top \nabla^2 f(x_k) G_{(j)} / \|G_{(j)}\|^2 \right\}$, with $a_j = 0$ for any $j \neq \ell$, again satisfies (5.2) and allows to minimize the gap between $\lambda_{\min}\left[\nabla^2 f(x_k)\right]$ and $\lambda_{\min}^{(k)}\left[\nabla^2 f(x_k)\right]$ (see Lemma 5.4). Moreover, it basically encompasses also the choice for the negative curvature direction adopted in [21]. Furthermore, observe that the choice of the sequence $\{a_j\}$ in (5.1) is only claimed to fulfill (5.2), in order to provide a negative curvature direction $s_k = z/\|z\|$ satisfying either Proposition 5.1 or Proposition 5.5.

Let us consider a Truncated Newton method for solving (1.1); in case $s_k$ were either used in a curvilinear framework (i.e. combined with a Newton–type direction) or as a stand alone search direction, then it is expected to be at least a *descent direction* and possibly a *gradient–related* one. Then, recalling [6, Section 3.1], we have sufficient conditions for the choice of the test within the Bunch–Kaufmann decomposition, so that the approximate solution $\bar{d}_k$ of (2.1) is gradient–related. Furthermore, by Propositions 3.1 and 3.2 of [6] it is possible to show that also the vector $a_j G_{(j)}$ in (5.1) is gradient–related (see also Remark 5.2), provided that it is chosen exactly as the vectors $uu$ and $vv$ in the Reverse–Scheme of [6]. Indeed, broadly speaking, the vector $a_j G_{(j)}$ in (5.1) is equivalently obtained by reducing the Lanczos process/Bunch–Kaufmann procedure used in SYMMBK to a CG method. Thus, the proper choice of the coefficient $a_j$ makes the vector $a_j G_{(j)}$ of descent for $f(x)$ at $x_k$. In this regard, note that the

parameter $\omega$ in the test within the Bunch–Kaufmann decomposition in [6], yet affects also the computation of the negative curvature direction $s_k$, through the coefficients $\{a_j\}$. Therefore, on the overall the vector $a_j G_{(j)}$ is gradient–related, provided that (recalling the Reverse–Scheme in [6]) the coefficient $a_j$ is chosen so that $a_j G_{(j)} \equiv uu$ or $a_j G_{(j)} \equiv vv$.

Hence, under mild assumptions on the sequence $\{a_j\}$ in (5.1), the vector $s_k$ can be safely and fully used within Truncated Newton methods, to guarantee convergence towards stationary points satisfying also second order necessary optimality conditions.

Finally, as suggested in Remark 5.2, since $\|G_k^{-1}\|$ is bounded, then also the vector $z$ in (5.1) (with the positions (5.2)) may be considered an alternative negative curvature direction fulfilling (i)–(iii) in (2.5). In the numerical experimentation we adopt the negative curvature direction defined in (5.1), to cope with the well known drawbacks related to carelessly combining the Newton type direction $d_k$ and a negative curvature direction, when they show a large difference between their norms. Indeed, generally $d_k$ is not expected to have unit norm (in this regard see the detailed discussion reported in the next Section 6).

# 6 Numerical experiments

To assess the performances of a Truncated Newton method which uses negative curvature directions computed according to the described approach, we carried out an extensive numerical testing. We considered the same optimization framework adopted in [6], namely a Truncated Newton method based on the SYMMBK procedure (implemented in the routine HSL_MI02 of the HSL Mathematical Software Library [28]) to solve the Newton equation. In [6], SYMMBK pivoting rule has been slightly modified in order to provide, at each outer iteration $k$, a gradient–related Newton–type direction $d_k$. The reader can refer to [6] for any detail.

Now, as described in the previous sections, we exploit the SYMMBK routine also for iteratively computing, at each outer iteration $k$, a negative curvature direction $s_k$ (if any). Then, we implemented the iterative scheme (1.4) where the steplength $\alpha_k$ is computed through the standard curvilinear linesearch procedure in [31]. We are aware (see, e.g., [26]) of the already mentioned problem which arises when combining a Newton-type direction and a negative curvature direction in the scheme (1.4), namely the possible different scaling of the two directions that could lead to inefficiency of the algorithm (as we also observed in preliminary testing). However, here we do not propose any strategy for possibly selecting the best promising direction between the two, as also detailed in literature (see, e.g., [13, 21, 26, 35] and the discussion regarding this issue included in the Introduction). This is motivated by the fact that we are focusing on an assessment of the approach we propose, namely the effectiveness of the negative curvature direction computed via SYMMBK method.

Moreover, an additional safeguard must be considered in dealing with negative curvature directions within a Truncated Newton method. Indeed, when the iterates approach a local minimizer (hence the Newton-type direction entails a superlinear convergence rate), the use of negative curvature directions might partially "spoil" such

good convergence rate, imposing a tight exploitation of local geometries associated with the function topology.

To overcome these two drawbacks, in the light of the conclusions proposed in Section 5.1 and Remark 5.2, we adopted the next two *zeroing rules* for the negative curvature direction: at each outer iteration $k$, a negative curvature direction $z$ (if any) is ignored when

– its norm consistently differs from the norm of $d_k$, namely

$$\|z\| > \eta_1 \|d_k\| \quad \text{or} \quad \|z\| < \eta_2 \|d_k\|, \qquad \eta_1 > \eta_2 > 0; \tag{6.1}$$

– it results

$$\|\nabla f(x_k)\| < \gamma_1 \qquad \text{and} \qquad \frac{z^\top \nabla^2 f(x_k) z}{\|z\|^2} > -\gamma_2, \qquad \gamma_1 > 0, \gamma_2 > 0. \tag{6.2}$$

The rationale behind the first rule (based on (6.1)) relies on the fact that, computing a negative curvature direction $z$ as described in Section 5 implies that both $d_k$ and $z$ are built using the same conjugate directions (the columns of the matrix $G_k$ in (4.4)). Hence, one would expect that both the vectors $z$ and $d_k$ have a similar scaling. If this does not occur, then the scaling problem possibly arises: we believe that the computed negative curvature direction could introduce a detrimental effect on the efficiency of the algorithm, hence we do not consider it.

The second zeroing rule (based on (6.2)), concerns the situation that may occur whenever the algorithm generates iterates sufficiently close to a second order critical point, i.e. a stationary point where the Hessian matrix is nearly positive semidefinite, and thus the Rayleigh–Ritz quotient (along $z$) is negative and close to zero. Also in this case, we do not consider the contribution of the negative curvature direction.

As concerns the values of the parameters adopted in (6.1) and (6.2), in our experimentation we use the following: $\eta_1 = 10^2, \eta_2 = 10^{-2}$ and $\gamma_1 = 10^{-3}, \gamma_2 = 10^{-2}$. More sophisticated strategies could be certainly adopted and will be the subject of future work (see also [23]).

Regarding the truncation criterion of the inner iterations, we adopt the standard *residual based criterion* [16], namely $\|\nabla f(x_k) + \nabla^2 f(x_k) d_k\| \leq \eta_k \|\nabla f(x_k)\|$, where $\eta_k = \min\left(\|\nabla f(x_k)\|, \frac{\sqrt{n}}{k}\right)$ is the forcing function we use. For the stopping criterion of the algorithm (outer iterations) we use the standard one $\|\nabla f(x_k)\| \leq 10^{-5} \max(1, \|x_k\|)$. Finally, we state that an algorithm fails to solve a problem if 3600 seconds of CPU time limit is exceeded.

To perform an extensive numerical testing, in our experimentation we considered all the large scale unconstrained test problems from CUTEst collection [27], amounting to 166 test problems, with sizes in the range 1,000–10,000. All the runs were performed on a PC with Intel Core i7-4790K CPU @ 4.00GHz with 32 Gb RAM.

The results consider the number of outer iterations, the number of function evaluations, the number of inner iterations, the optimal function value and the CPU time. Moreover, in the case an algorithm incorporates negative curvature directions, their number is also considered. For the sake of brevity, we report in the following only a

summary of the obtained results and we refer the reader to [22] for all the detailed complete results.

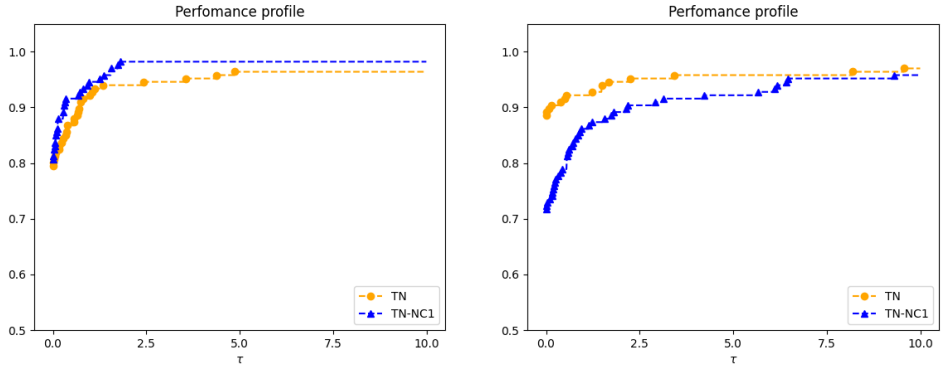
First, we ran on the whole test set the algorithm which does not consider negative curvature directions that we named TN. The obtained results will be used as benchmark in the subsequent comparisons.

### 6.1 Use of the negative curvature direction (5.1)

Now, we consider the algorithm named TN-NC1 which uses the negative curvature direction defined in (5.1), adopting the zeroing rule in (6.1) and (6.2). We compare the performance of TN-NC1 and TN algorithms by using the widely adopted *performance profiles* [18]. Figures 1a, 1b, 1c and 1d report such performance profiles in terms of number of outer iterations, number of function evaluations, number of inner iterations and CPU time, respectively. These plots refer to the whole set of test problems. They clearly highlight the efficiency and the robustness in terms of iterations and inner iterations of the algorithm TN-NC1, with respect to the algorithm TN. On the other hand, TN-NC1 algorithm on the overall requires a larger number of function evaluations. Actually, this behaviour was expected, since it is due to the standard curvilinear linesearch procedure used which is based on a rough combination of the two search directions. We are convinced that a more sophisticated linesearch technique might be adopted, when second order information related to negative curvature directions is available. In this regard, a further investigation seems mandatory in the light of the outcomes of the present numerical experiences. As regards CPU time, the corresponding profile shows a modest increase of the time required by algorithm TN-NC1.

The detailed results (see [22]) provide us with some further interesting evidences. First, it can be observed that on 9 difficult test problems both the algorithms fail to converge within 3600 seconds of CPU time. On 1 test problem (CURLY30 with $n = 10,000$) the use of even a few negative curvature directions (say 5) enables the algorithm TN-NC1 to converge in only 129.58 seconds, whereas TN fails to converge. On 71 test problems negative curvature directions are actually encountered and used by TN-NC1 algorithm; on 30 of these test problems, the two algorithms converge to different local minimizers. Table 1 reports the optimal function values obtained by TN and TN-NC1 algorithms on the latter test problems. This table clearly highlights the expected capability of the algorithm TN-NC1 to converge towards better local minimizers. The cases in which a better value is obtained by TN algorithm are comparatively very few. From the detailed results it can be also observed that, in many cases, the additional effort due to the computation of negative curvature directions, is balanced by a greater overall efficiency of the algorithm, so that CPU time required by TN-NC1 algorithm on average shows only a modest increase.

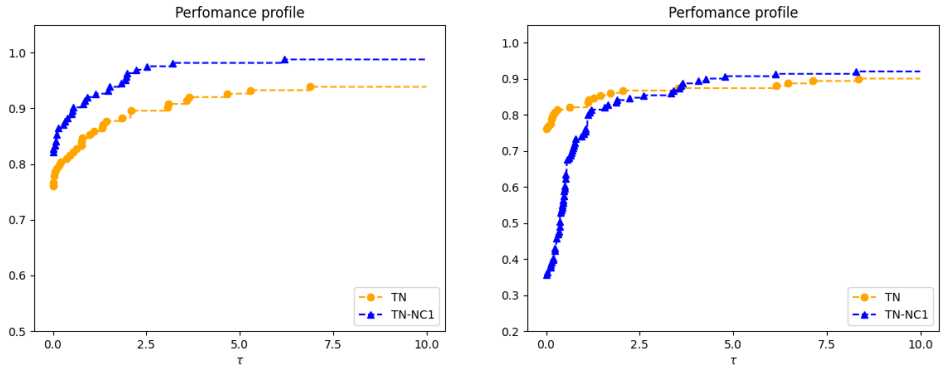
Since on a number of test problems the two algorithms converge towards different minimizers, in a second experiment we again plot performance profiles comparing TN and TN-NC1 algorithms but considering only those test problems where they converge to the same local minimizer. Figures 2a, 2b, 2c and 2d report these plots in terms of number of outer iterations, number of function evaluations, number of inner iterations and CPU time, respectively. A comparison of these new plots with the ones in Figure 1




(a) *Outer iterations.*




(b) *Function evaluations.*




(c) *Inner iterations.*




(d) *CPU time.*

**Fig. 1**: Performance profiles for the *whole set* of 166 test problems.

leads to an important consideration regarding the use of the performance profiles when ranking different algorithms. Indeed, in terms of number of function evaluations and CPU time a better performance of TN algorithm is observed both in Figures 1b, 2b, and in Figures 1d, 2d. Conversely, in terms of number of outer iterations (see Figure 1a and Figure 2a) and in terms of number of inner iterations (see Figure 1c and Figure 2c) we carry out a different conclusion: the profiles referred to the whole test set do not seem to agree with those related to test functions where both the algorithms converge to the same local minimizer. On the other hand, one could reasonably claim that including in the performance comparison test problems where the algorithms converge towards different local minimizers could be unfair. Actually, the key point is that, in comparing the performance among different algorithms, performance profiles do not take into account the "quality" of the solution found by different algorithms, i.e., their capability to determine better local minimizers. This drawback could be overcome neither by adopting *performance profiles* nor by using the *data profiles* proposed in [36].

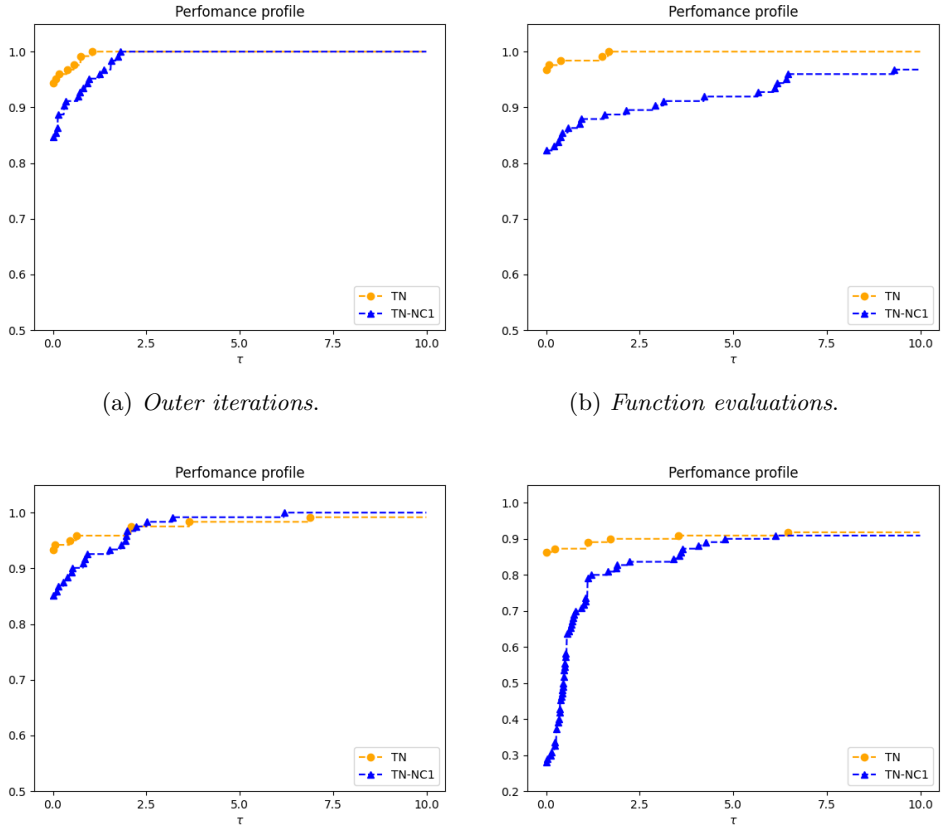
**Table 1**: Optimal function values obtained by TN and TN-NC1 algorithms, on those test problems where they converge to different local minimizers. For each problem the best function value obtained is highlighted in bold.

| *Problem* | *n* | TN algorithm *function value* | TN-NC1 algorithm *function value* |
|---|---|---|---|
| BROYDN7D | 1000 | 5.598036E+02 | **3.047159E+02** |
| BROYDN7D | 5000 | 3.125637E+03 | **1.659976E+03** |
| BROYDN7D | 10000 | **1.233247E+00** | 6.883895E+03 |
| CHAINWOO | 1000 | 4.324322E+02 | **2.514980E+02** |
| CHAINWOO | 4000 | 2.294394E+03 | **1.582515E+03** |
| CHAINWOO | 10000 | 6.305963E+03 | **2.890864E+03** |
| COSINE | 1000 | -9.985153E+02 | **-9.990000E+02** |
| CURLY10 | 1000 | -9.765683E+04 | **-1.003125E+05** |
| CURLY10 | 5000 | -4.665618E+05 | **-5.015815E+05** |
| CURLY10 | 10000 | -1.002761E+06 | **-1.003163E+06** |
| CURLY20 | 1000 | -9.814893E+04 | **-1.003093E+05** |
| CURLY20 | 5000 | -4.844972E+05 | **-5.015758E+05** |
| CURLY20 | 10000 | -1.002861E+06 | **-1.003162E+06** |
| CURLY30 | 1000 | -9.854450E+04 | **-1.000507E+05** |
| CURLY30 | 5000 | -4.929491E+05 | **-5.015808E+05** |
| FLETCBV3 | 1000 | **-1.083664E+05** | -3.189503E+03 |
| FLETCBV3 | 5000 | -2.221834E+07 | **-6.880147E+07** |
| FLETCBV3 | 10000 | **-1.321473E+09** | -1.147714E+09 |
| GENHUMPS | 1000 | 3.252751E+02 | **2.359907D-10** |
| NCB20 | 1010 | 9.663298E+02 | **9.208174E+02** |
| NCB20 | 5010 | -1.394735E+03 | **-1.447533E+03** |
| NCB20 | 10010 | -2.642929E+03 | **-5.313355E+03** |
| NONCVXUN | 1000 | 2.405825E+03 | **2.327616E+03** |
| NONCVXU2 | 1000 | 2.381367E+03 | **2.317103E+03** |
| SINQUAD | 10000 | -2.642227E+07 | **-2.642315E+07** |
| SPARSINE | 1000 | 6.341091E+05 | **1.618616E+05** |
| SPARSINE | 5000 | **1.567241E+07** | 1.697258E+07 |
| SPARSINE | 10000 | **6.448147E+07** | 7.005234E+07 |
| SPMSRTLS | 1000 | 6.321999E+01 | **5.608497E-02** |
| SPMSRTLS | 10000 | 2.937016E+00 | **3.409363E-11** |

This motivates the use, in the current paper, of novel profiles based on the "quality" of the solution provided by the algorithms introduced in [22]. Of course, this new tool appears of fundamental importance for comparing algorithms which use negative curvature directions versus algorithms that do not use them. In the next Section 6.3, after briefly recalling the definition of the *quality profiles* we use them for ranking different algorithms.

## 6.2 Use of alternative negative curvature directions

The numerical experimentation reported in the previous Section 6.1 relies on the use of the negative curvature direction given by (5.1). However, to enhance our investigation on negative curvature directions within Truncated Newton methods it might be worth considering two alternatives for computing such directions. In place of considering the sum in (5.1), we might select only one column of the matrix $G_{(j)}$ (along with the associated coefficient $a_j$). In particular, it is possible to consider in (5.1) the index set




(a) *Outer iterations.* (b) *Function evaluations.*

(c) *Inner iterations.* (d) *CPU time.*

**Fig. 2**: Performance profiles for test problems where both the algorithms TN and TN-NC1 converge to the *same local minimizer* (136 test problems).

$\mathcal{J}$ such that $|\mathcal{J}| = 1$, being possibly

$$z = a_{\bar{h}} G_{(\bar{h})}, \quad \text{where} \quad \bar{h} = \operatorname*{argmin}_h \{\mu_h \mid \mu_h < 0\}, \tag{6.3}$$

and

$$z = a_{\bar{l}} G_{(\bar{l})}, \quad \text{where} \quad \bar{l} = \min_h \{h \mid \mu_h < 0\}. \tag{6.4}$$

The rationale behind the choice in (6.3) is to consider only the smallest negative eigenvalue of the matrix $D_k$ in (4.3). The selection (6.4) consists of considering only the first negative eigenvalue of $D_k$. We name by TN-NC2 and TN-NC3 the algorithms which use the negative curvature directions in (6.3) and (6.4), respectively. In both the cases, we adopt again the zeroing rules in (6.1) and (6.2) for negative curvature directions, too. Once again, the complete results are included in [22] and here we

only report a summary. In particular, to assess the capability of the algorithms which incorporate negative curvature directions to determine better local minimizers, in the next Section 6.3 we use the novel *quality profiles* (see [22]). The aim is to compare the effectiveness of the negative curvature directions adopted in TN-NC1, TN-NC2 and TN-NC3 versus TN.

## 6.3 Benchmarking via *Quality Profiles*

As we already pointed out, when ranking different algorithms which use negative curvature directions, it is very important to assess the quality of the obtained optimal solution. Indeed, they could have different capability of locating local minimizers with lower value of the objective function. To this aim, a novel class of profiles named *Quality Profiles* has been proposed in [22]. Now we briefly recall it and refer the reader to [22] for any detail.

Assume the set $\mathcal{S}$ of iterative solvers and the set $\mathcal{P}$ of test problems are considered, being $|\mathcal{S}| \geq 2$ and $|\mathcal{P}| \geq 1$ their cardinalities. Let $x_0^{(p)} \in \mathbb{R}^n$ be the starting point for all the solvers on the test problem $p \in \mathcal{P}$, and let $f^{(p)}(x)$ be the objective function of the problem $p \in \mathcal{P}$. Let $f_L^{(p)}$ be a reference value for the objective function of the problem $p$. If $x^*$ indicates the best iterate found by the solver $s \in \mathcal{S}$ on the test problem $p$ (for the sake of simplicity we drop in $x^*$ the dependency on both $s$ and $p$), different choices are possible for $f_L^{(p)}$ (see [22] for some examples). Here we adopt

$$f_L^{(p)} = \min_{s \in \mathcal{S}} \left\{ f_s^{(p)}(x^*) \right\},$$

where $f_s^{(p)}(x^*)$ denotes the optimal function value determined by the solver $s$ on the problem $p$. Then, for each solver $s \in \mathcal{S}$ we consider the ratio

$$Q_s(\tau) = \frac{1}{|\mathcal{P}|} size \left\{ p \in \mathcal{P} \ : \ f_s^{(p)}(x^*) - f_L^{(p)} \leq \tau \left[ f^{(p)}(x_0^{(p)}) - f_L^{(p)} \right] \right\}, \tag{6.5}$$

being $\tau \geq 0$. The *Quality Profile* is the collection of the plots of all the ratios $\{Q_s(\tau)\}$ in (6.5) when[4] $\tau \in [0, 1]$. Quality profiles inherit from *performance profiles* in [18] and from *data profiles* in [36] several interesting properties. Moreover, they enjoy some additional features fundamental for thoroughly ranking different algorithms (see [22] where quality profiles have been introduced).

In Figure 3 we report the quality profiles comparing the algorithms TN-NC1 and TN. Plots in Figure 3 confirm that, as expected, in terms of quality of the solution found, the algorithm TN-NC1 outperforms the algorithm TN.

Moreover, in the spirit of carrying out the investigation on the use of negative curvature directions, to enhance the capability of an algorithm to determine better local minimizers, we now include in the comparison also the results obtained by algorithms TN-NC2, TN-NC3 described in Section 6.2 (which use alternative negative curvature

[4] Since for any solver $s$ it results $f_s^{(p)}(x^*) \leq f^{(p)}(x_0^{(p)})$, then for any $s \in \mathcal{S}$ we have $Q_s(\tau) = 1$ for all $\tau \geq 1$. Hence it suffices to consider for the parameter $\tau$ the range $[0, 1]$.

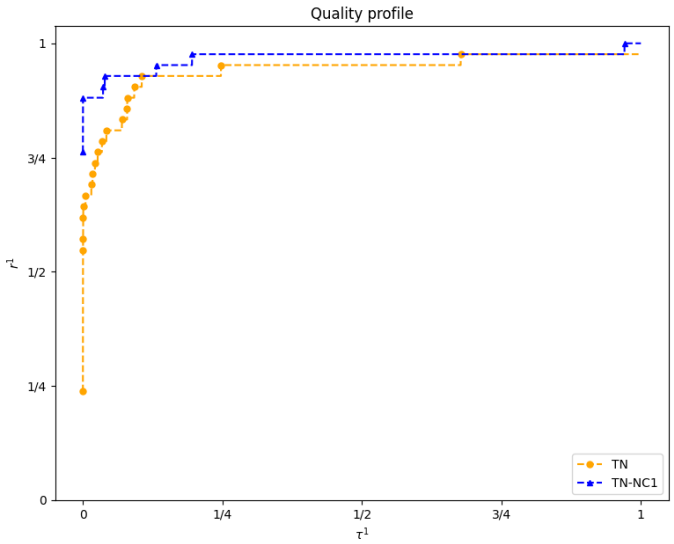



**Fig. 3**: Quality profile for the two solvers TN and TN-NC1.

directions). Figure 4 reports a comparison, adopting quality profiles, for the four different algorithms TN, TN-NC1, TN-NC2 and TN-NC3. First observe that not all the four tracks in the last figure converge to the same value (i.e. one) when $\tau = 1$. This can indeed be easily explained by recalling that the piece of information related to the number of failures associated to each solver is duly taken into account by quality profiles (the last consideration can be regarded as a counterpart of a similar property of performance and data profiles). On the overall, the plots in Figure 4 confirm the fact that algorithms which use negative curvature directions in most cases determine better local minimizers. Moreover, TN-NC3 seems the most effective among the three codes which incorporate negative curvature directions. We recall that in this last algorithm the negative curvature direction is computed only selecting information associated to the first negative eigenvalue of the diagonal matrix $D_k$ in (3.3). The last fact confirms what was also pointed out in [21], and reveals that in the large scale settings the accurate and expensive computation of a negative curvature direction can be possibly dodged.

We finally observe that, it is possible to enhance the quality profiles in order to expand or compress portions of the plots, for better comparing solvers when tracks are very close. We believe that there is no further room to include this extension here. We only refer the interested reader to [22].

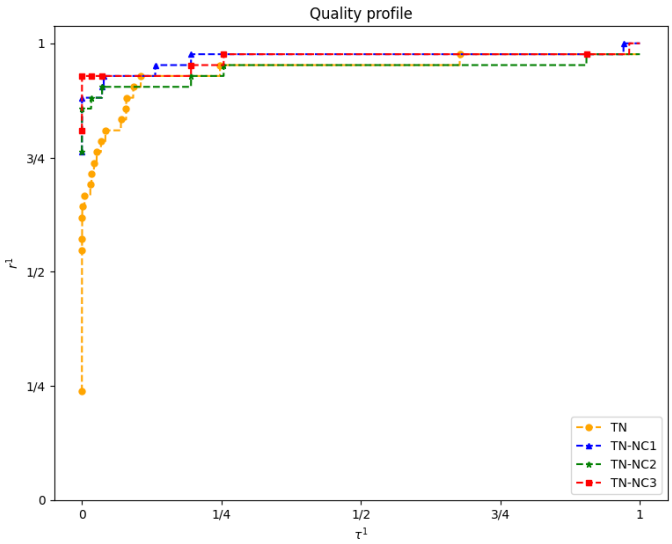



**Fig. 4**: Quality profile for the four solvers TN, TN-NC1, TN-NC2 and TN-NC3.

# 7 Conclusions

Following the recent rekindled interest in using negative curvature directions in non-convex optimization frameworks, in this paper we investigate on negative curvature directions within Truncated Newton methods. In large scale settings, the use of such directions is often dodged due to the computational effort required for generating appropriate negative curvature directions; indeed, many approaches are often based on matrix storage/factorization or they need the computation of eigenvalues of the Hessian matrix. On the other hand, the adoption of appropriate negative curvature directions, guaranteeing convergence towards second order critical points, allows an algorithm not to get stuck at saddle points.

Here we fruitfully exploited the SYMMBK method for iteratively computing effective negative curvature directions, only requiring a minimal additional effort. This allowed us to integrate such directions within the Truncated Newton method proposed in [6]. As a result we obtain an algorithm with enhanced capability to determine better local minimizers. The extensive numerical experimentation clearly evidenced this important feature, and thanks to a novel benchmarking tool, namely the quality profiles, we were able to fully show it.

As concluding remark, we believe that this paper provides evidence of the benefit of incorporating negative curvature directions. This meets (at least partially) the specific issue pointed out in the recent literature (see, e.g., the Introduction of [13]).

**Acknowledgments.** G. Fasano thanks the National Research Council–Marine Technology Research Institute (CNR-INSEAN), Italy. G. Fasano and M. Roma

are grateful to the working group GNCS of IN$\delta$AM (Istituto Nazionale di Alta Matematica), Italy, for the support they received.

# Declarations

- The authors have no conflict of interest.
- Data availability: no relevant data are used in the paper. All the complete results supporting the analysis in the article are available in the Technical Report [22] that can be accessed through the reported link.

# References

[1] Avelino, C.P., Moguerza, J.M., Olivares, A., Prieto, F.J.: Combining and scaling descent and negative curvature directions. Mathematical Programming **128**, 285–319 (2011)

[2] Baldi, P., Hornik, K.: Neural networks and principal component analysis: Learning from examples without local minima. Neural networks **2**, 53–58 (1989)

[3] Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large–scale machine learning. SIAM Review **60**, 223–311 (2018)

[4] Bray, A.J., Dean, D.S.: Statistics of critical points of Gaussian fields on large-dimensional spaces. Physical review letters **98**, 150201 (2007)

[5] Bunch, J.R., Kaufman, L.C.: Some stable methods for calculating inertia and solving symmetric linear equations. Mathematics of Computations **31**, 163–179 (1977)

[6] Caliciotti, A., Fasano, G., Potra, F., Roma, M.: Issues on the use of a modified Bunch and Kaufman decomposition for large scale Newton's equation. Computational Optimization and Applications **77**, 627–651 (2020)

[7] Caliciotti, A., Fasano, G., Roma, M.: Novel preconditioners based on quasi–Newton updates for nonlinear conjugate gradient methods. Optimization Letters **11**, 835–853 (2017)

[8] Caliciotti, A., Fasano, G., Roma, M.: Preconditioned nonlinear conjugate gradient methods based on a modified secant equation. Applied Mathematics and Computation **318**, 196–214 (2018)

[9] Chandra, R.: Conjugate gradient methods for partial differential equations. PhD thesis, Yale University, New Haven (1978). Research Report 129

[10] Choromanska, A., Henaff, M., Mathieu, M., Arous, G.B., LeCun, Y.: The loss surfaces of multilayer networks. In: Artificial Intelligence and Statistics, pp. 192–204 (2015). PMLR

[11] Conn, A.R., Gould, N.I.M., Toint, P.L.: Trust–region Methods. MPS–SIAM Series on Optimization, Philadelphia, PA (2000)

[12] Cullum, J.K., Willoughby, R.A.: Lanczos Algorithms for Large Symmetric Eigenvalue Computations. Birkhauser, Boston (1985)

[13] Curtis, F.E., Robinson, D.P.: Exploiting negative curvature in deterministic and stochastic optimization. Mathematical Programming **176**, 69–94 (2019)

[14] Dauphin, Y.N., Pascanu, R., Gulcehre, C., Cho, K., Ganguli, S., Bengio, Y.: Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. Advances in neural information processing systems **27** (2014)

[15] De Leone, R., Fasano, G., Roma, M., Sergeyev, Y.D.: Iterative grossone-based computation of negative curvature directions in large-scale optimization. Journal of Optimization Theory and Applications **186**, 554–589 (2020)

[16] Dembo, R.S., Eisenstat, S.C., Steihaug, T.: Inexact Newton methods. SIAM Journal on Numerical Analysis **19**, 400–408 (1982)

[17] Dembo, R.S., Steihaug, T.: Truncated-Newton algorithms for large-scale unconstrained optimization. Mathematical Programming **26**, 190–212 (1983)

[18] Dolan, E.D., Moré, J.: Benchmarking optimization software with performance profiles. Mathematical Programming **91**, 201–213 (2002)

[19] Fasano, G.: Planar–conjugate gradient algorithm for large–scale unconstrained optimization, Part 1: Theory. Journal of Optimization Theory and Applications **125**, 523–541 (2005)

[20] Fasano, G.: Planar–conjugate gradient algorithm for large–scale unconstrained optimization, Part 2: Application. Journal of Optimization Theory and Applications **125**, 543–558 (2005)

[21] Fasano, G., Lucidi, S.: A nonmonotone truncated Newton-Krylov method exploiting negative curvature directions, for large scale unconstrained optimization. Optimization Letters **3**, 521–535 (2009)

[22] Fasano, G., Piermarini, C., Roma, M.: Exploiting SYMMBK method for the full computation of negative curvature directions. Technical Report 06-2023, Dipartimento di Ingegneria Informatica, Automatica e Gestionale "A. Ruberti", SAPIENZA Università di Roma (2023). http://users.diag.uniroma1.it/biblioteca/it/node/6133

[23] Fasano, G., Piermarini, C., Roma, M.: Bridging the gap between trust–region methods (TRMs) and linesearch based methods (LBMs) for nonlinear programming: Quadratic sub–problems. Department of Management, Università Ca'

Foscari Venezia, Working Paper (8) (2022)

[24] Fasano, G., Roma, M.: Iterative computation of negative curvature directions in large scale optimization. Computational Optimization and Applications **38**, 81–104 (2007)

[25] Ferris, M.C., Lucidi, S., Roma, M.: Nonmonotone curvilinear linesearch methods for unconstrained optimization. Computational Optimization and Applications **6**, 117–136 (1996)

[26] Gould, N.I.M., Lucidi, S., Roma, M., Toint, P.L.: Exploiting negative curvature directions in linesearch methods for unconstrained optimization. Optimization Methods and Software **14**, 75–98 (2000)

[27] Gould, N.I.M., Orban, D., Toint, P.L.: CUTEst: a constrained and unconstrained testing environment with safe threads. Computational Optimization and Applications **60**, 545–557 (2015)

[28] HSL 2013: A collection of Fortran codes for large scale scientific computation. http://www.hsl.rl.ac.uk/

[29] Jiang, H., Robinson, D.P., Vidal, R., You, C.: A nonconvex formulation for low rank subspace clustering: algorithms and convergence analysis. Computational Optimization and Applications **70**, 395–418 (2018)

[30] Lucidi, S., Rochetich, F., Roma, M.: Curvilinear stabilization techniques for truncated Newton methods in large scale unconstrained optimization. SIAM Journal on Optimization **8**, 916–939 (1998)

[31] McCormick, G.P.: A modification of Armijo's step-size rule for negative curvature. Mathematical Programming **13**, 111–115 (1977)

[32] Moré, J.J., Sorensen, D.C.: On the use of directions of negative curvature in a modified Newton method. Mathematical Programming **16**, 1–20 (1979)

[33] Nash, S.G.: A survey of truncated-Newton methods. Journal of Computational and Applied Mathematics **124**, 45–59 (2000)

[34] Nash, S.G.: Newton-type minimization via the Lanczos method. SIAM Journal on Numerical Analysis **21**, 770–788 (1984)

[35] Olivares, A., Moguerza, J.M., Prieto, F.J.: Nonconvex optimization using negative curvature within a modified linesearch. European Journal of Operational Research **189**, 706–722 (2008)

[36] Wild, S., Moré, J.: Benchmarking derivative–free optimization algorithms. SIAM J. Optimization **20**, 172–191 (2009)